

# Assignment Report: Inventory Management System

Arshpreet Singh Sidhu (Student ID: 815805)

**Deadline: 11:59PM on 1 March 2024**

## 1. Introduction:

This assignment is about the development of a Python inventory management program. The program offers functionalities such as adding items to the inventory, removing items from the inventory, and displaying the current inventory through a menu-based interface.

## 2. Problem Statement:

You are tasked with creating a Python program to manage a small inventory for a store. Your program should allow users to perform various actions such as adding items to the inventory, removing items from the inventory, and displaying the current inventory. Error handling for invalid inputs and edge cases is also required.

## 3. Methodology:

- **User Interaction Design:** The program features a user-friendly interface with clear prompts and a menu-based system for easy navigation and minimizing user errors.

```
while not break_loop:
    choice = input("Enter your Choice: ")
    msg = ""
    match choice:
        case "1":
            msg = add_item(items)
        case "2":
            msg = remove_item(items)
        case "3":
            display_inventory(items)
        case "4":
            clear_screen()
            print("Exiting Program.....\n")
            break_loop = True
        case default:
            msg = "Please choose a valid option!"

    if not break_loop and choice != "3":
        welcome()
        print(msg + "\n")
```

- **Modularization:** The code is modularized using functions to handle each functionality separately, ensuring a clear and organized structure.

```
def clear_screen(): ...  
  
def welcome(): ...  
  
def add_item(inventory): ...  
  
def remove_item(inventory): ...  
  
def display_inventory(inventory): ...
```

- **Error Handling:** Appropriate error-handling techniques are used to manage invalid inputs and edge cases. For instance, In function for adding items to inventory, conditional statements are used to verify the range of quantity inputted by the user, and try-except blocks are used to handle cases where the user enters incorrect input types such as strings or floats for value of “quantity”.

```
def add_item(inventory):  
    name = input("Enter item name: ").lower()  
  
    try:  
        quantity = int(input("Enter quantity: "))  
  
        if quantity < 0:  
            return "Quantity can not be negative, Use 'Remove an item' option  
to reduce items"  
        if quantity == 0:  
            return "Quantity not be zero"  
  
        inventory[name] = quantity if name not in inventory else  
inventory[name] + quantity  
        return f'{name.capitalize()} ({quantity}) added to the inventory.'  
    except:  
        return "Quantity can only be in integers"
```

- **Pattern Matching with “match case”:** The match case statement is used to handle different cases based on user input, instead of traditional if-else statements. The use of match case increases the clarity and consistency of the code, making it easier to understand and maintain. It also contributes the Pythonic principle of readability and simplicity. For instance, when the user selects an option from the menu, instead of using multiple if-else statements to check each possible choice, a match case statement can be used to handle each option more efficiently.

```

match choice:
    case "1":
        msg = add_item(items)
    case "2":
        msg = remove_item(items)
    case "3":
        display_inventory(items)
    case "4":
        clear_screen()
        print("Exiting Program.....\n")
        break_loop = True
    case default:
        msg = "Please choose a valid option!"

```

## 4. Implementation:

The implementation involves the use of functions and the “os” module to add items, remove items, display the inventory, clear the console screen, and handle user inputs. There are multiple functions made to modularize the code.

In the `clear_screen()` function, the “command” variable is set to “cls” or “clear” according to the operating system that the user uses; it will be “cls” if the operating system is Windows, otherwise, it will be “clear” for other operating systems because the command to clear the screen varies according to the operating system. Then the command gets executed on the command line of the user using the “`os.system()`” function from the “os” module to clear the console screen.

In the `welcome()` function, the main menu text is displayed after clearing the console screen by using the `clear_screen()` function.

In the `add_item()` function, a dictionary “inventory” is passed to the function. The user is asked to enter the name of the new item using the input function, then it is stored in the variable name. After that, a try-except block is used because when the user is asked to enter the quantity of that same item, if they enter anything else than numbers, an error will occur, and the except block will display an error to them instead of crashing the program. Then edge cases of quantity are handled, i.e., if the user enters 0 or any negative numbers. If everything goes right till here, then it will add the item to the inventory variable passed to the function using index brackets “[ ]” and a success message is displayed.

In the `remove_item()` function, handling edge cases and input validation is similar to the `add_item()` function. If everything goes right, it will remove the item from the inventory dictionary passed to it; if the user wants to remove everything, it will be removed using the “`.pop()`” method; otherwise, it will be subtracted using index brackets “[ ]”.

In the `display_inventory()` function, all the items of inventory objects are displayed on the console using “string formatting” to display everything in a tabular form.

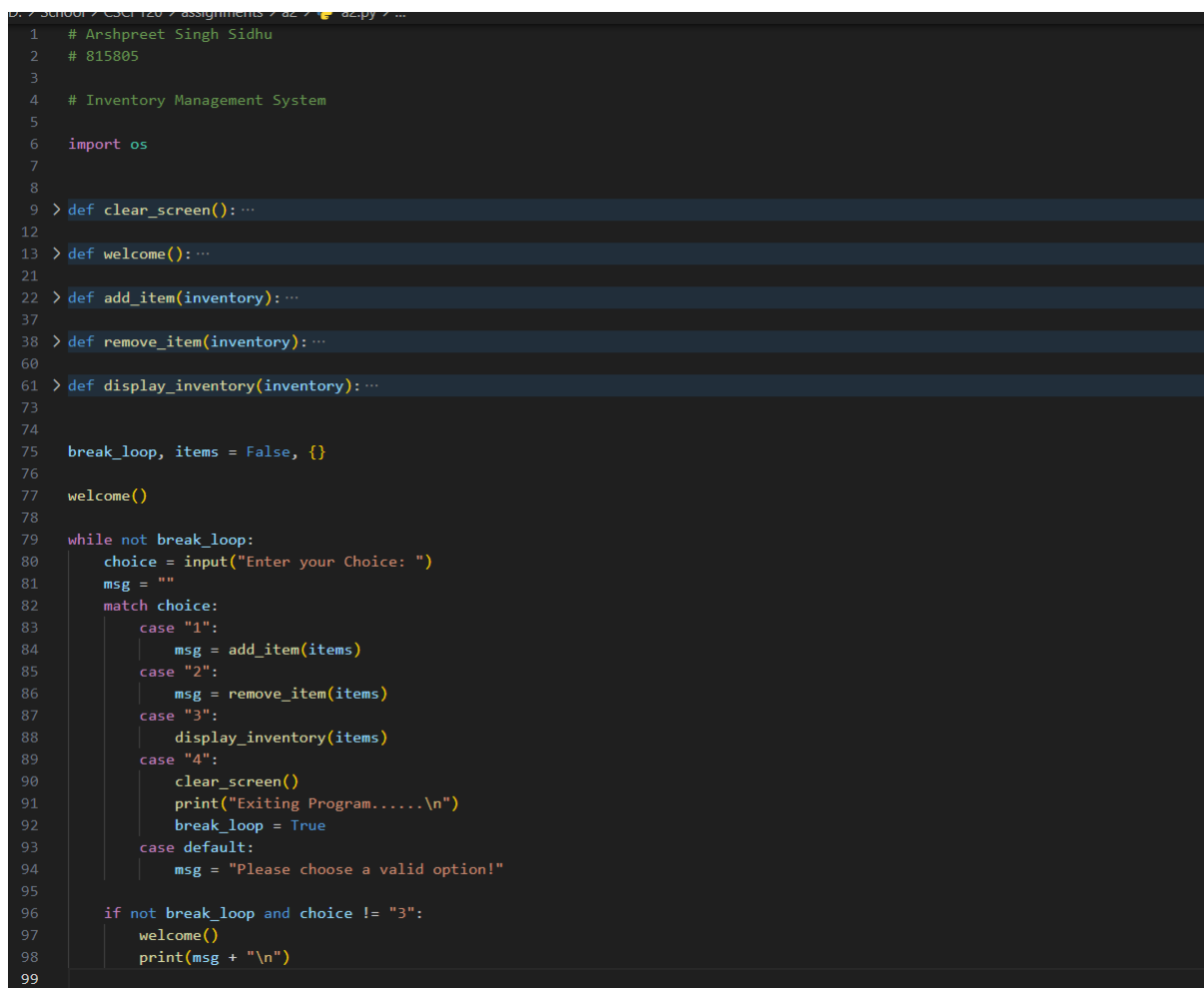
After these function declarations, the main code begins with initializing variables used, i.e., `break_loop` and `items`. `break_loop` is used to terminate the program when the user wants to. The “items” variable is used to store inventory. Then the welcome message is displayed using the `welcome()` function, and the loop of the program starts. In the loop, the user is asked to make a choice according to the menu displayed in the

welcome() function. Then a match-case is used to check what the user chooses, and a “case default” is used to handle edge cases. Then functions are called according to the user’s choice. If the user pressed “4”, then it exits the loop by changing the break\_loop variable to true. If not, it saves the return of the function to a variable “msg” to display the output of the function. After the match-case, the welcome message and the message from the function are displayed.

## 5. Results:

The program manages the inventory effectively, allowing users to add, remove, and view items in the inventory. Error handling increases the stability of the program even when user enters invalid inputs.

## 6. Screenshots:

A screenshot of a code editor showing the source code of an inventory management system. The code is written in Python and includes comments for the author's name and ID, the program title, and imports. It defines several functions: clear\_screen(), welcome(), add\_item(), remove\_item(), and display\_inventory(). The main logic is in a while loop that prompts the user for a choice and uses a match-case statement to call the appropriate function. A break\_loop variable is used to control the loop, and a default case handles invalid inputs. The code is numbered from 1 to 99.

```
1 # Anshpreet Singh Sidhu
2 # 815805
3
4 # Inventory Management System
5
6 import os
7
8
9 > def clear_screen(): ...
12
13 > def welcome(): ...
21
22 > def add_item(inventory): ...
37
38 > def remove_item(inventory): ...
60
61 > def display_inventory(inventory): ...
73
74
75 break_loop, items = False, {}
76
77 welcome()
78
79 while not break_loop:
80     choice = input("Enter your Choice: ")
81     msg = ""
82     match choice:
83         case "1":
84             msg = add_item(items)
85         case "2":
86             msg = remove_item(items)
87         case "3":
88             display_inventory(items)
89         case "4":
90             clear_screen()
91             print("Exiting Program.....\n")
92             break_loop = True
93         case default:
94             msg = "Please choose a valid option!"
95
96 if not break_loop and choice != "3":
97     welcome()
98     print(msg + "\n")
99
```

Figure 1: Screenshot of the source code of the program.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Welcome to Inventory Management System!

1. Add an item
2. Remove an item
3. Display current inventory.
4.Exit

=====
Item Names | Quantity
-----
Brinjal   | 10
Banana    | 69
=====

Enter your Choice: █
```

Figure 2: Screenshot of the execution of the program.

## 7. Efficiency Approaches:

- **Use of While Loops:** While loops are used to create a menu-based interface that allows users to interact with the inventory system repeatedly until they choose to exit. This ensures a good user experience without the need to restart the program after each operation.
- **Match Case for User Input:** The `match case` statement is employed to handle user input effectively. By using `match case`, the code becomes more structured and easier to maintain compared to traditional `if-else` statements.
- **Case Insensitivity:** Case insensitivity is used to make the program more user-friendly. User input for item names is converted to lowercase before saving so that the program can process all type in casing.
- **String Formatting Operations for Display:** String formatting operations are used to present the inventory in a well-organized and readable format. The `%-12s` format specifier is used to align item names and quantities neatly in columns.
- **Use of clear\_screen():** The `clear\_screen()` function is used to clear the terminal screen before displaying the menu or executing any operation. This makes a clean and clutter-free interface, so that previous outputs don't cluttering the display.

## 8. References:

Online resources during the development of this assignment are Official Python Documentation, W3Schools, Lecture Slides, provided by Jeetho Ryoo